FACULTY OF TECHNOLOGY AND ENGINEERING

DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY AND RESEARCH

DEPARTMENT OF COMPUTER ENGINEERING

A.Y. 2023-24 [ODD]

LAB MANUAL

CE260: MICROPROCESSOR AND COMPUTER ORGANIZATION

Semester: III **Academic Year: 2023**

Subject Code: CE260 **Subject Name: Microprocessor and Computer Organization**

Student Id: 22DCE006 **Student Name: PROBIN BHAGCHANDANI**

PRACTICAL INDEX

Sr. No.	AIM	Assigned Date	Completion Date	Grade	Assessment Date	Signature
1	Generation and Types of Computer.					
2	Write a program to convert a given number system to other number system.					
3	Implement a circuit in Logisim to display given binary number in decimal on to seven segment display.					
4	Implement a circuit in Logisim which perform Addition and Subtraction.					
5	Write a program which perform multiplication using booth algorithm.					
6	1. Add and Subtract the 16-bit number in memory locations 4000H and 4001H to the 16-bit number in memory locations 4002H and 4003H. The most significant eight bits of the two numbers to be added are in memory locations 4001H and 4003H. Store the result in memory locations 4004H and 4005H with the most significant byte in memory location 4005H. 2. Write a program to multiply & divide the number stored at 4000H by 08H and store the result at 4001H & 4002H. 3. Write an assembly language program to convert temperature in F to C. C=(F-32) * 5/9					
7	Consider two 8-bit data are stored					

	at memory location 4001H and 4002H. perform following logical operation on it and store result from 4002H location. 1. OR Operation 2. AND Operation 3. NOT Operation 4. XOR Operation 5. Logical Left & Right Shift 6. Arithmetic Left and Right Shift 7. Rotate Left and Right with Carry 8. Rotate Left and Right without Carry					
8	1. Calculate the sum of series of numbers from the memory location 4000H & store the result at 400AH location. 2. Modify above the program such a way that it halts the execution if carry generated & stores the intermediate result at 400AH location. 3. Write an assembly language program to find the no. of odd numbers and even numbers, given an array of n numbers.					
9	Find out whether the given string is palindrome or not and print appropriate message. Don't use procedure.					
10	1. Write an assembly language program to find the largest number in an array. 2. Write an assembly language program to factorial of the given Number.					
11	Write an assembly language program to arrange an array of data in ascending order. The length of the list is at memory location 2200H and the series itself begins from memory location 2201H					

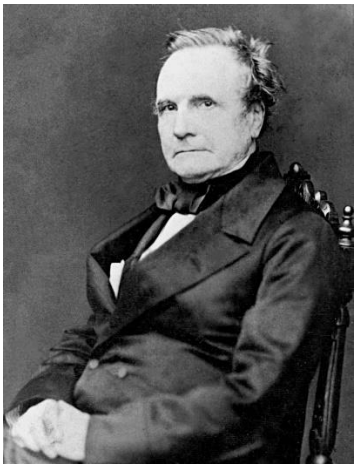
PRACTICAL-1

AIM: Generation and Types of Computer.

A computer is a machine or device that performs processes, calculations and operations based on instructions provided by a software or hardware program. It has the ability to accept data (input), process it, and then produce outputs.FULL FORM OF COMPUTER

C=Common
O=Operating
M=Machine
P=Particularly
U=Used for
T=Technological and
E=Engineering
R=Research

Initially, the computer was introduced to solve complex mathematical problems but later on by innovation, it is used for multi-purpose**. The computer was invented by Charles Babbage in 1822. He made the first mechanical computer. Charles Babbage is known as the "Father of Computing ".**



CHARLES BABBAGE

Applications of Computer:
1)Entertainment
2)Education
3)Banking
4)Designing

5)Government facilities
6)ART
7)Medical Industry and many more.

Types of Computer-based on size and speed

1) Micro Computer – They are single CPU single user systems which are used at home and shops and schools.
They are further categorized into 5 categories: Personal, Workstation, Laptops, Mobiles/Tablets, Embedded Computers

i) Personal – They are used for common applications like browsing, learning, gaming, business, etc.
ii)Workstation- They have a more powerful CPU and have higher memory they are comparatively more expensive and are used for niche work like designing, animation, and complex mathematical calculations.
iii)Laptops – They are portable computers. They are also known as notebook computers
iv) Mobiles/Tablets – They are used for entertainment, gaming, as camera,etc.

2) Mini Computer- They are also known as mid range servers They lie between mainframe and microcomputers they are multi-users systems they have more storage and memory . They are used as webservers , database servers, gaming servers eg-VAX, Magnum,etc.

MINI COMPUTER

3) Mainframe Computer – They are designed to handle huge volumes of data and infromation they perform multiple tasks from users in parallel theyalso have multiple processors and are fast and expensive they are used by large organcization like airports

corporate buildings , colleges,etc. eg-IBM UNIVAC,etc,

MAINFRAME COMPUTER

4) Super Computer- They are the fastest and most expensive large and have multiple cpus so they can handle many instructions in parallel used for complex calculation weather forecasting nuclear science rocket launching eg- summit , sierra , deep blue , param

SUPERCOMPUTER

Generations of Computer

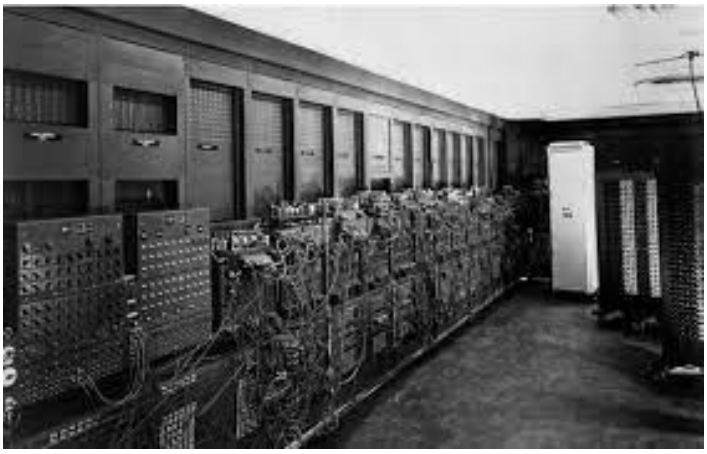
The word generation refers to how the hardware and software has evovled over the years.

1st Generation- Vaccum Tubes

(1945 to 1956)

- **It had thousand of vaccum tubes .**
- **It consumes lot of power .**
- **It generated a large amount of heat .**
- **This excessive heat release was the reason for their breakdown.**
- **It's structures were very large in size.**
- **They were very heavy in weight.**
- **Its size was that like of a room.**
- **They were very expensive.**
- **For pragamming they used machine language using at punched cards and paper tapes.**
- **They were not portable.**
- **They were fastest calculating devices their time**
- **They were able to do 5000additions/second and were able to do 350multiplications / second.**

Eg-ENIAC , EDVAC , UNIVAC-1 , IBM-701 , IBM-650 .



ENIAC

2nd Generation- Transistors

(1956 to 1963)

- **It had transistors.**
- **They were smaller in design and faster than 1st generation .**
- **They were cheaper and more reliable .**
- **They were more energy efficient .**
- **They still generated a lot of heat .**
- **They used magnetic core technology to store the instructions in memory and used machine and assembly language to program.**

Eg-IBM7090 , CDC3600, UNIVAC1108 , CDC3600.

CDC3600

3rd Generation- Integrated Circuits

(1964 to 1971)

- **It used lntegrated Circuits (ICs)**
- **They were smaller in stucture and faster.**
- **They were cheaper than 2nd generation computer.**
- **Multiprogramming operating systems were used for these computers.**
- **They were accessed by using keyboard ,monitor.Low maintenance .**
- **It even had a opertaing system to enable it to perform**
- **eg-IBM360 , PDP11**

IBM360

4th Generation- Very large scale Integrations

(1971 to 1980)

- **They had microprocessor with VLSI ie.**
- **Very large scale Integrations. It had five thousands (5000) transistors and other connecting elements on a single silicon chip.**
- **They were small and portable.**
- **They were cheapest and used to work at high speed with high accuracy and reliability.**
- **They had larger memory.**
- **They were portable.**
- **They were used for GUI(Graphical User Interface) application software and handheld devices.**

Eg-windows computer , mac computers, laptops,etc.

5th Generation- Ultra large scale Integrations

(1981 to now)

- **They will be based on Artificial Intelligence AI .**
- **They will use ULSI ie. Ultra large scale Integrations**
- **So making it cheaper and fastest and also self reliant .**
- **They will use Quantum technology and Nano technology which will make it very efficient and fastest among all types .**
- **They are known as the Intelligent Computers.**

CONCLUSION: From this practical, we learned about the history of computer, the basics of the computer, different types of computers and different generations of the compute .

Staff Signature:
Grade:
Remarks by the Staff:

PRACTICAL-2

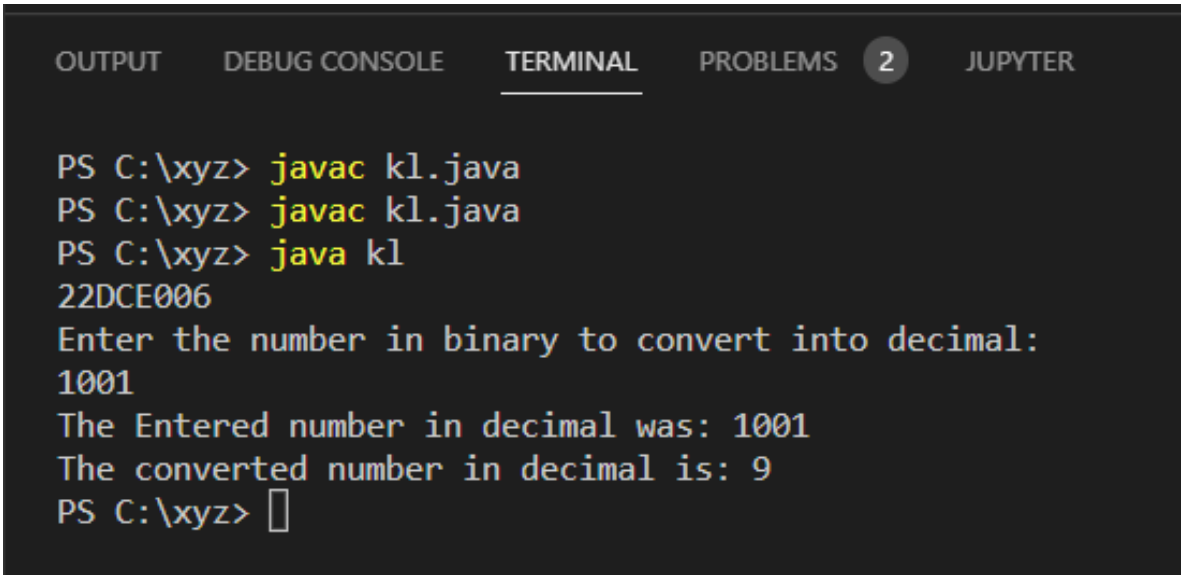
AIM: Write a program to convert a given number system to other number system.

PROGRAM CODE:

```
import java.util.*;
public class kl
{
        public static void main(String args[])
        {
                Scanner sc=new Scanner(System.in);
                int dec_num=0,rem,bin_num=0, base=1;
                int trial_number;
                System.out.println(x: "22DCE006");
                System.out.println(x: "Enter the number in binary to convert
                into decimal: ");
                bin_num=sc.nextInt ();
                trial_number=bin_num;
                while(trial_number>0)
                {
                        rem=trial_number%10 ;
                        dec_num= dec_num+rem*base;
                        trial_number=trial_number / 10 ;
```

```
                base-base*2;
            }
            System.out.println("The Entered number in decimal was: "+bin_num);
            System.out.println("The converted number in decimal is: "+dec_num);
        }
}
```

OUTPUT:



CONCLUSION:

From this practical we understood the logic of conversion of numbers from binary number system into decimal number system. The conversion of numbers from binary to decimal is important as it helps to read numbers that are represented as a set of 0s and 1s.

$(\text{Decimal Number})_{10} = (d0 \times 2^0) + (d1 \times 2^1) + (d2 \times 2^2) + \ldots + (dn\text{-}1 \times 2^{n-1})$

Staff Signature:

Grade:

Remarks by the Staff:

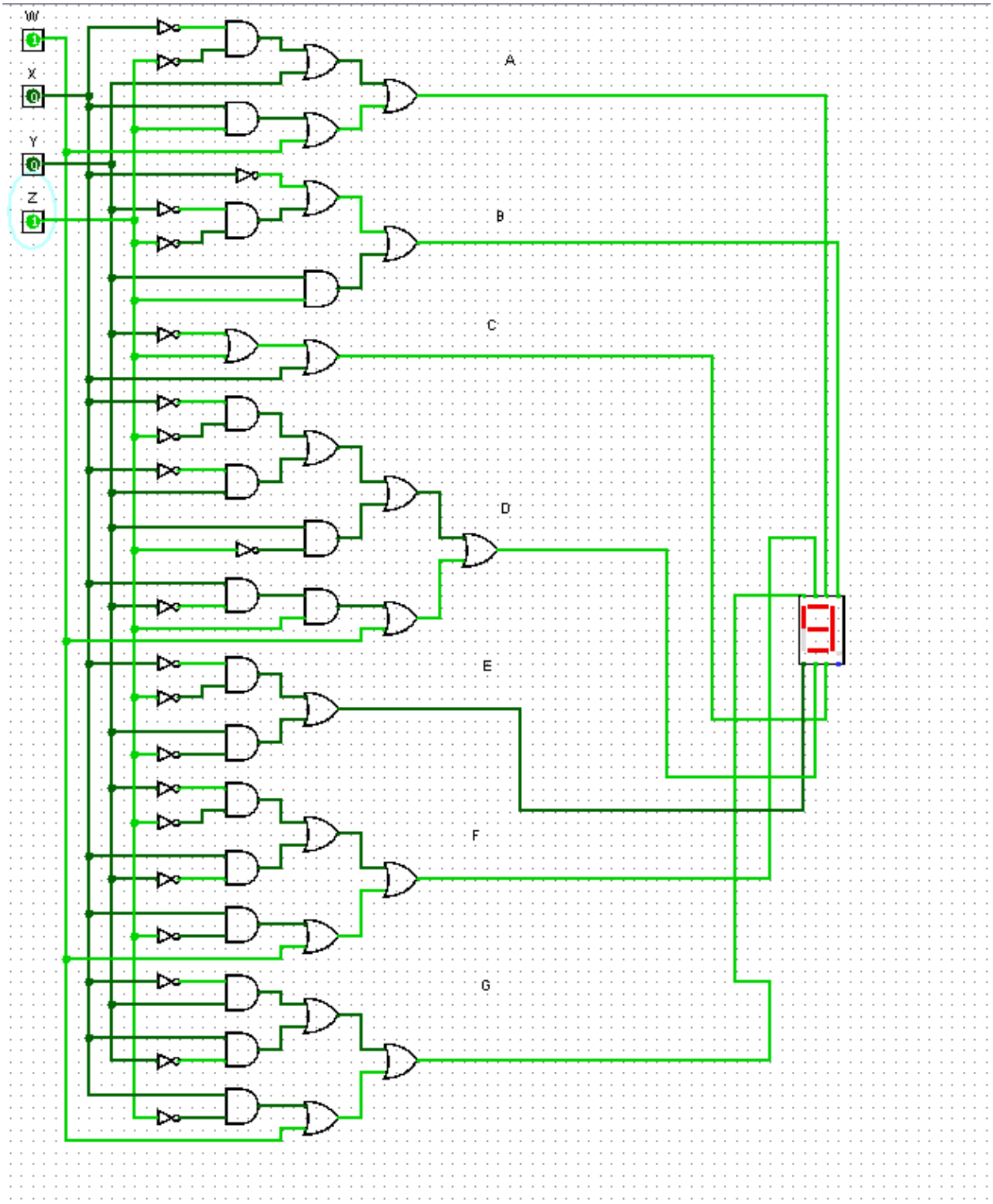
PRACTICAL-3

AIM: Implement a circuit in Logisim to display given binary number in decimal on to seven segment display.

TRUTH TABLE:

W	X	Y	Z	A	B	C	D	E	F	G
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	x	x	x	x	x	x	x
1	0	1	1	x	x	x	x	x	x	x
1	1	0	0	x	x	x	x	x	x	x
1	1	0	1	x	x	x	x	x	x	x
1	1	1	0	x	x	x	x	x	x	x
1	1	1	1	x	x	x	x	x	x	x

Circuit Diagram:


W
X
Y
Z
A
B
C
D
E
F
G

CONCLUSION: Seven Segment display is commonly used in electronic display devices for decimal numbers and in some cases, basic characters. It is made of seven different illuminating segments which are arranged in a way to form numbers and characters by selecting different combinations of segments.

Staff Signature:

Grade:

Remarks by the Staff:

PRACTICAL-4

AIM: Implement a circuit in Logisim which perform Addition and Subtraction.

PROCEDURAL APPROACH:

Step-1) Take two 4-bit input switches, one for input A and the other for input B.

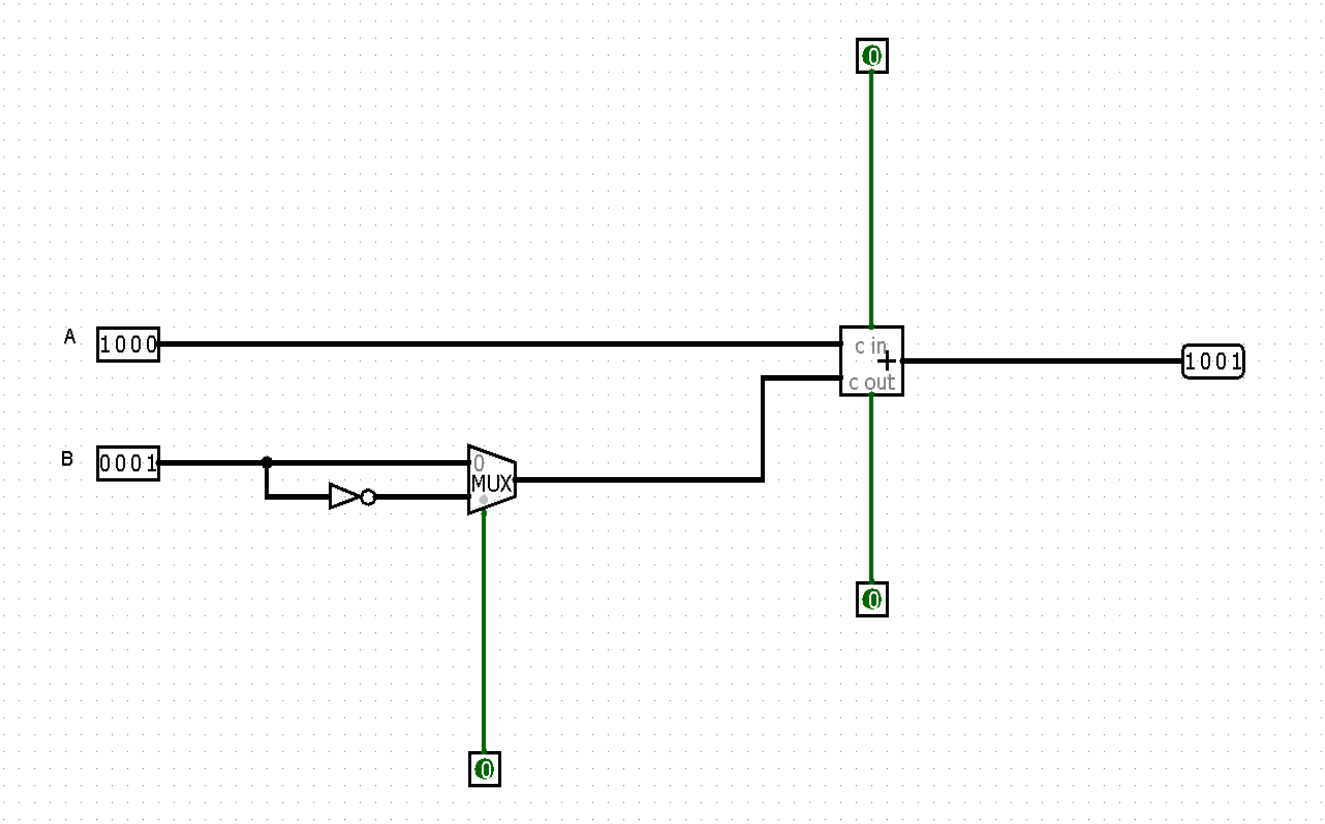
Step-2) Next take a multiplexer from the arithmetic folder and set B and B' or (B bar) as its inputs followed by a select line for the Multiplexer which will select the input.

Step-3) For operation take a full adder from the arithmetic folder and take inputs as A and output from multiplexer which had inputs B and B’.

Step-4) Enter initial carry manually.

Step-5) For output display 4-Bit key pin is used which shows the output in 4-Bits.

Circuit Diagram:-



CONCLUSION: From this practical, I learned the implementation of addition and subtraction operations using Multiplexer and Full-Adder.

Staff Signature:

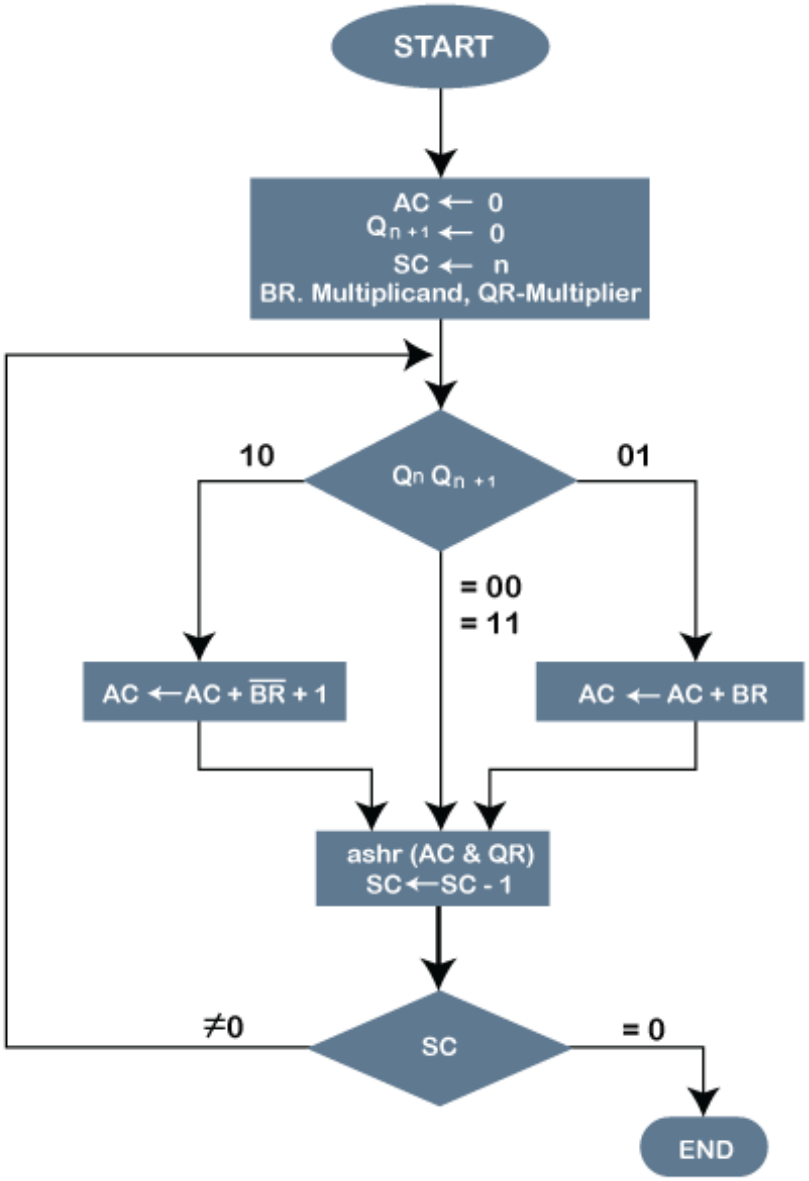
Grade:

Remarks by the Staff:

PRACTICAL-5

AIM: Write a program which perform multiplication using Booth Algorithm.

Algorithm:



PROGRAM CODE:

```
public class example
{
   public int multiply(int n1, int n2)
   {
      int[] m = binary(n1);
```

```
        int[] m1 = binary(-n1);
        int[] r = binary(n2);
        int[] A = new int[9];
        int[] S = new int[9];
        int[] P = new int[9];
        for (int i = 0; i < 4; i++)
        {
            A[i] = m[i];
            S[i] = m1[i];
            P[i + 4] = r[i];
        }
        display(A, 'A');
        display(S, 'S');
        display(P, 'P');
        System.out.println();

        for (int i = 0; i < 4; i++)
        {
            if (P[7] == 0 && P[8] == 0);
            else if (P[7] == 1 && P[8] == 0)
                add(P, S);
            else if (P[7] == 0 && P[8] == 1)
                add(P, A);
            else if (P[7] == 1 && P[8] == 1);
            rightShift(P);
            display(P, 'P');
        }
        return getDecimal(P);
    }
    public int getDecimal(int[] B)
    {
        int p = 0;
        int t = 1;
        for (int i = 7; i >= 0; i--, t *= 2)
            p += (B[i] * t);
        if (p > 64)
            p = -(256 - p);
        return p;
    }
    public void rightShift(int[] A)
    {
        for (int i = 8; i >= 1; i--)
            A[i] = A[i - 1];
    }
    public void add(int[] A, int[] B)
```

```
    {
        int carry = 0;
        for (int i = 8; i >= 0; i--)
        {
            int temp = A[i] + B[i] + carry;
            A[i] = temp % 2;
            carry = temp / 2;
        }
    }
    public int[] binary(int n)
    {
        int[] bin = new int[4];
        int ctr = 3;
        int num = n;
        if (n < 0)
            num = 16 + n;
        while (num != 0)
        {
            bin[ctr--] = num % 2;
            num /= 2;
        }
        return bin;
    }
    public void display(int[] P, char ch)
    {
        System.out.print("\n"+ ch +" : ");
        for (int i = 0; i < P.length; i++)
        {
            if (i == 4)
                System.out.print(" ");
            if (i == 8)
                System.out.print(" ");
            System.out.print(P[i]);
        }
    }
    public static void main (String[] args)
    {
        System.out.println("\n22DCE006\nBooth Algorithm Test\n");
        example b = new example();
        int n1=2 , n2=3;
        int result = b.multiply(2,3);
        System.out.println("\n\nResult : "+ n1 +" * "+ n2 +" = "+ result);
    }
}
```

Output:

```
PS D:\Probin's Work\Java Programming> javac example.java
PS D:\Probin's Work\Java Programming> java example

22DCE006
Booth Algorithm Test


A : 0010 0000 0
S : 1110 0000 0
P : 0000 0011 0

P : 1111 0001 1
P : 1111 1000 1
P : 0000 1100 0
P : 0000 0110 0

Result : 2 * 3 = 6
PS D:\Probin's Work\Java Programming>
```

CONCLUSION: From this practical, I learned about the Multiplication of two Signed 2's Complement binary numbers using Booth Algorithm and its working.

Staff Signature:

Grade:

Remarks by the Staff:

PRACTICAL-6

AIM:

1. Add and Subtract the 16-bit number in memory locations 4000H and 4001H to the 16-bit number in memory locations 4002H and 4003H. The most significant eight bits of the two numbers to be added are in memory locations 4001H and 4003H. Store the result in memory locations 4004H and 4005H with the most significant byte in memory location 4005H.

2. Write a program to multiply & divide the number stored at 4000H by 08H and store the result at 4001H & 4002H.

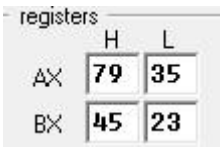
3. Write an assembly language program to convert temperature in F to C. C=(F-32) * 5/9

1)Code

ADD:

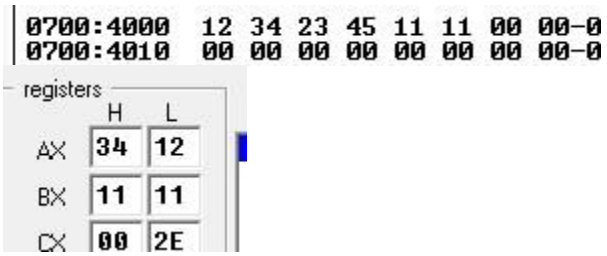
```
mov [4000h], 12H
mov [4001h],34H
mov [4002h], 23H
mov [4003h], 45H
mov AL,[4000h]
mov AH, [4001h]
mov BL, [4002h]
mov BH, [4003h]
add AX, BX
mov [4005H],AL
mov [4004H], AH
```

```
0700:4000  12 34 23 45 79 35 00 00-00 00 00
0700:4010  00 00 00 00 00 00 00 00-00 00 00
```



SUBTRACT:

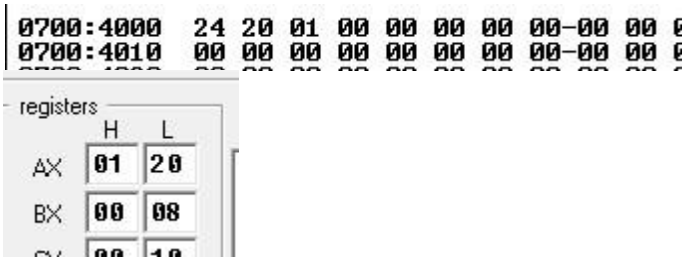
```
mov [4000h], 12H
mov [4001h], 34H
mov [4002h], 23H
mov [4003h], 45H
mov AL, [4000h]
mov AH, [4001h]
mov BL, [4002h]
mov BH, [4003h]
sub BX,AX
mov [4005H], BL
mov [4004H], BH
```



2)Code

MULTIPLY:

```
mov [4000H],24H
mov AL,[4000H]
mov BL,08H
mul BL
mov [4001H],AX
```



DIVIDE:

```
mov [4000H],24H
mov AL,[4000H]
mov BL,08H
div BL
mov [4002H], AL
```

3)Code

```
MOV BX,[4000H]
SUB BX,32
MOV AX,5
MUL BX
MOV BX,9
DIV BX
MOV [4002H], AX
RET
```

CONCLUSION: Assembly language is a low level programming language . Also , We learned many instructions of the software to perform various operations.

Staff Signature:

Grade:

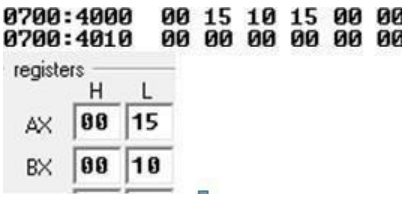
Remarks by the Staff:

PRACTICAL-7

AIM: Consider two 8-bit data are stored at memory location 4001H and 4002H. perform following logical operation on it and store result from 4002H location. 1. OR Operation 2. AND Operation 3. NOT Operation 4. XOR Operation 5. Logical Left & Right Shift 6. Arithmetic Left and Right Shift 7. Rotate Left and Right with Carry 8. Rotate Left and Right without Carry

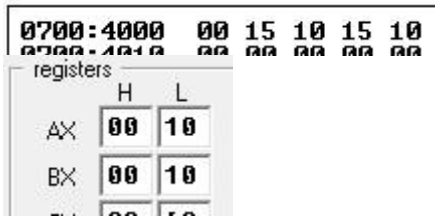
1) OR Operation

```
mov [4001h], 15h
mov [4002h], 10h
mov AL, [4001h]
mov BL, [4002h]
or AL, BL
mov [4003h], AL
```



2) AND Operation

```
mov AL, [4001h]
and AL, BL
mov [4004h], AL
```



3) NOT Operation

```
mov AL, [4001h]
not AL
mov [4005h], AL
```

```
0700:4000  00 15 10 15 10 EA 00
0700:4010  00 00 00 00 00 00 00 00
```

registers

	H	L
AX	00	EA
BX	00	10
CX	00	60

4) XOR Operation

```
mov AL, [4001h]
xorAL,BL
mov [4006h], AL
```

```
0700:4000  00 15 10 15 10 EA 05 00-00 00 00 00 00
0700:4010  00 00 00 00 00 00 00 00-00 00 00 00 00
```

registers

	H	L
AX	00	05
BX	00	10

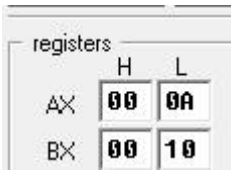
5) Logical Left & Right Shift

```
mov AL, [4001h]
shl AL,1
mov [4007h], AL
```

registers

	H	L
AX	00	2A
BX	00	10

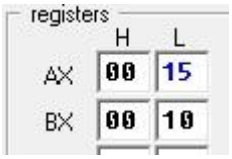
```
mov AL, [4001h]
shr AL,1
mov [4008h], AL
```



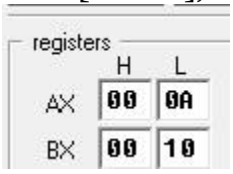
6) Arithmetic Left and Right Shift

```
mov AL, [4001h]
sal AL,1
mov [4009h], AL
```

```
0700:4000  00 15 10 15 10 EA 05 2A-0A 2A
0700:4010  00 00 00 00 00 00 00 00-00 00
```

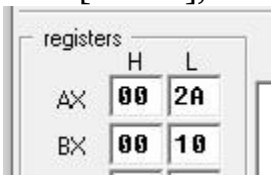



```
mov AL, [4001h]
sar AL,1
mov [4010h], AL
```



7) Rotate Left and Right with Carry

```
mov AL, [4001h]
rol AL,1
mov [4011h], AL
```



```
mov AL, [4001h]
ror  AL,1
mov [4012h], AL
```

```
0700:4000   00 15 10 15 10 EA 05 2A-0A 2A
0700:4010   0A 2A 8A 00 00 00 00 00-00 00
```

registers

	H	L
AX	00	8A
BX	00	10

8) Rotate Left and Right without Carry

```
ORG100h
MOV AL,[4001H]
ROL AL,1
MOV [4002H],AL
RET
```

registers

	H	L
AX	00	97
BX	00	00
CX	00	09
DX	00	00

```
ORG100h
MOV AL,[4001H]
ROR AL,1
MOV [4002H],AL
RET
```

registers

	H	L
AX	00	6F
BX	00	00
CX	00	09
DX	00	00

CONCLUSION: An instruction set is a group of commands for a central processing unit(CPU) in machine language. We learned about many arithmetic and logical operations with assembly language.

Staff Signature:

Grade:

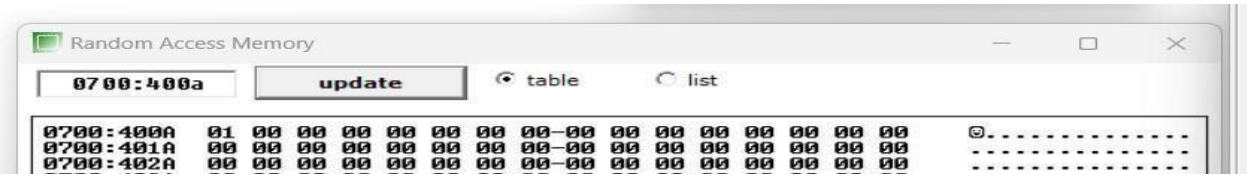
Remarks by the Staff:

PRACTICAL-8

AIM: 1) Calculate the sum of series of numbers from the memory location 4000H & store the result at 400AH location.

PROGRAM CODE:

```
org 100h
mov [4000h] , 01h
mov [4001h], 02h
mov [4002h] , 03h
mov [4003h] ,0Fah
 mov [4004h] ,  01h
mov si,4000h
 l1:mov al,[si]
add [400ah], al
inc sicmp si , 4005h
jne l1
ret
```

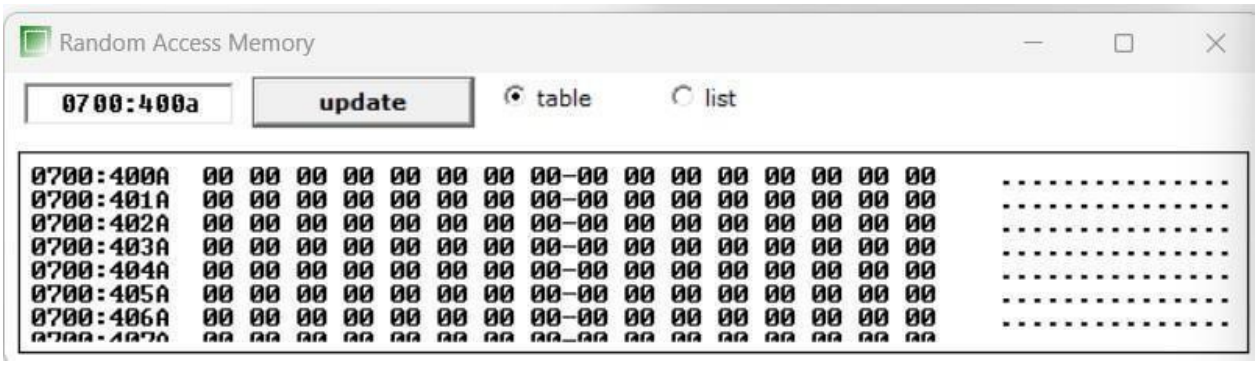


2) Modify above the program such a way that it halts the execution if carry generated &stores the intermediate result at 400AH location.

Program Code:

```
org 100h
mov [4000h] , 01h
mov [4001h], 02h
mov [4002h] , 03h
mov [4003h] ,0FAh
mov [4004h] ,  01h
mov si,4000h l1:
mov al,[si]
add [400ah], al
jc l2
inc si cmpsi , 4005
```

hjne
l1
l2:
RET

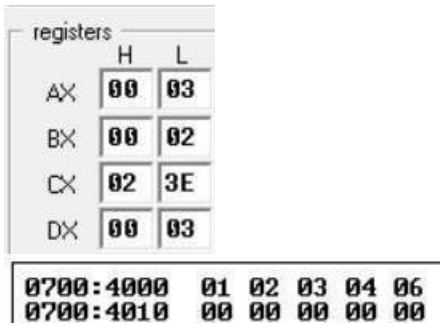


3) Write an assembly language program to find the no. of odd numbers and even numbers, givenan array of n numbers.

Program Code:

```
org 100h
mov [4000h] ,01h
mov [4001h] , 02h
mov [4002h] ,03h
mov [4003h] , 04h
mov [4004h] ,06h
mov bl , 02h
mov si , 4000h
l1: moval,[si]
div bl
cmp ah , 00
jz l2
inc ch
inc si cmp
si , 4005h
jne l1
jmp  l3
l2: inc dl
```

```
inc si
cmp si , 4005h
jne l1
jmp l3
l3:ret
```



Staff Signature:

Grade:

Remarks by the Staff:

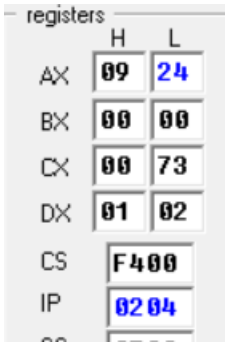
PRACTICAL-9

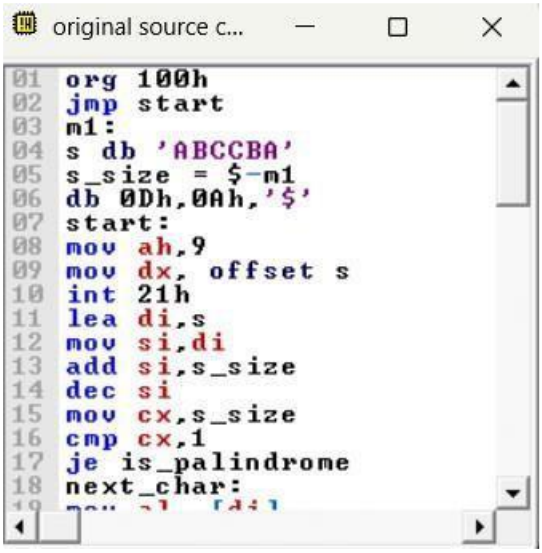
AIM: Find out whether the given string is palindrome or not and print appropriate message. Don't use procedure.

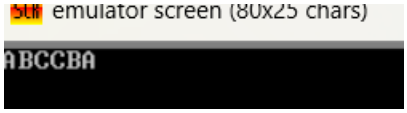
Program Code:

```
org 100hjmp startm1:
s db 'ABCCBA'
s_size = $-m1 db 0Dh,0Ah,'$'start:
mov ah,9
mov dx, offset sint 21h lea di,s mov si,di
add si,s_sizedec si mov cx,s_sizecmp cx,1
je is_palindrome next_char:
mov al, [di]mov bl, [si]cmp al,bl
jne not_palindromeinc di dec si
loop next_char
is_palindrome:mov ah,9
mov dx, offset msg1

int 21h jmp stop not_palindrome:mov ah,9

mov dx, offset msg2int 21h stop:
mov ah,0int 16h ret
msg1 db "this is palindrome?$" msg2 db "this is not a palindrome?$"ret
```

OUTPUT:







CONCLUSION: For a string to be a palindrome we need to compare the first character of the string with the last character and so on. I learned the comparison of characters using assembly language through emu8086.

Staff Signature:

Grade:

Remarks by the Staff:

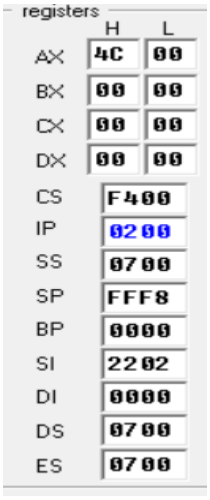
Practical 10

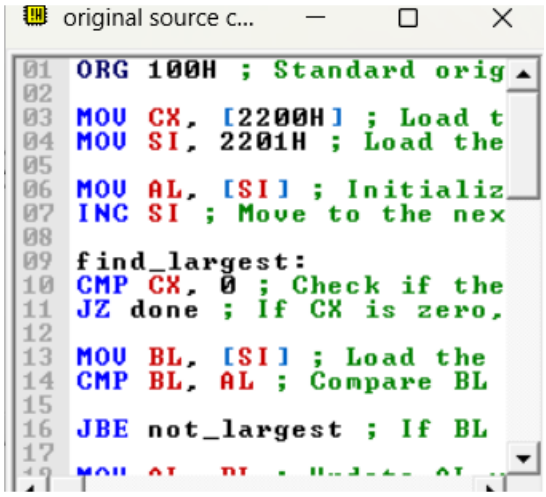
1.Write an assembly language program to find the largest number in an array.

Program code:

```
ORG 100H
MOV CX, [2200H] MOV SI, 2201H MOV AL, [SI] INC SI
find_largest:
CMP CX, 0
JZ done
MOV BL, [SI] CMP BL, AL
JBE not_largest MOV AL, BL
INC SI DEC CX
JMP find_largest
MOV AH, 4CH ; DOS exit function
INT 21H ; Call DOS to terminate the program
RET ; Return to DOS
```

OUTPUT:



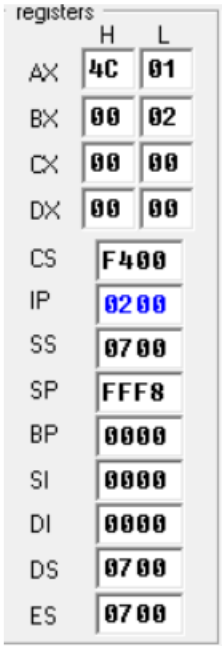


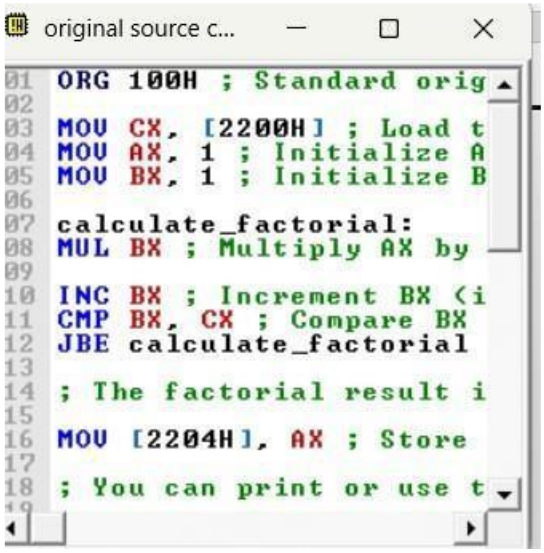
2.Write an assembly language program to factorial of the given Number.

Program Code:

```
ORG 100H ; Standard origin point for COM programs
MOV CX, [2200H] ; Load the input number into CX
MOV AX, 1 ; Initialize AX to 1 (for the factorial)
MOV BX, 1 ; Initialize BX to 1 (counter) calculate_factorial:
MUL BX ; Multiply AX by BX (AX = AX * BX)
 INC BX ; Increment BX (increment the counter)
CMP BX, CX ; Compare BX with the input number
JBE calculate_factorial ;
Jump if BX <= CX (continue loop if not done)
; The factorial result is now in AX
MOV [2204H], AX ; Store the result at memory location 2204H;
You can print or use the result here MOV AH, 4CH ; DOS exit function
INT 21H ; Call DOS to terminate the program
RET ; Return to DOS
```

OUTPUT:





Conclusion: From this practical I learned to find factorial of any number using assembly language through emu8086.

Staff Signature:

Grade:

Remarks by the Staff:

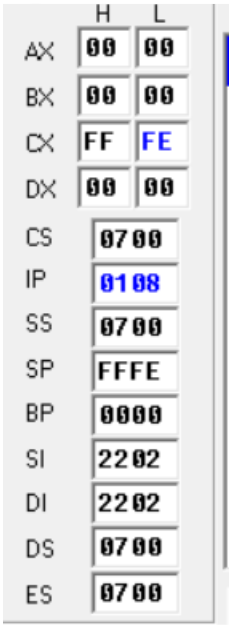
Practical 11

Write an assembly language program to arrange an array of data in ascending order. The length of the list is at memory location 2200H and the series itself begins from memory location 2201H.

Program Code:

```
ORG 100H
MOV CX, [2200H] DEC CX
MOV SI, 2201H
SI: element
MOV DI, SI
MOV AL, [DI] into AL
INC DI element
MOV BH, [DI] into BH
CMP AL, BH
JAE no_swap
; Swap AL and BH
MOV [DI], AL
DEC DI
MOV [DI], BH
no_swap:
INC SI ; Move to the next pair of elements
LOOP outer_loop ; Continue until CX becomes zero
; The list is now sorted in ascending order
MOV AH, 4CH ;
DOS exit function
INT 21H ; Call DOS to terminate the program
RET ; Return to DOS
```

OUTPUT:




```
ORG 100H ; Standard orig

MOV CX, [2200H] ; Load t
DEC CX ; Decrement CX to

MOV SI, 2201H ; Load the

outer_loop:
MOV DI, SI ; Set DI to p
MOV AL, [DI] ; Load the
INC DI ; Move DI to poin
MOV BH, [DI] ; Load the

CMP AL, BH ; Compare AL
JAE no_swap ; If AL >= B

; Swap AL and BH
MOV [DI], AL
```

Conclusion: From this practical I learned to sort the data of the arrays in descending manner using assembly language through emu8086.

Staff Signature:

Grade:

Remarks by the Staff: